

A simple algorithm for the evaluation of the hypergeometric series using quasi-linear time and linear space

S.V.Yakhontov

Saint Petersburg State University
Faculty of Mathematics and Mechanics
sergey_home_mail@inbox.ru

June 14, 2011

Abstract

A simple algorithm with time complexity $O(M(n)\log(n)^2)$ and space complexity $O(n)$ for the evaluation of the hypergeometric series with rational coefficients is constructed ($M(n)$ being the complexity of integer multiplication). It is shown that this algorithm is suitable in practical informatics for constructive analogues of often used constants of analysis.

Introduction. In this paper we construct an algorithm for the calculation of the approximate values of the hypergeometric series with rational coefficients whose implementation is simple and which is quasi-linear in time and linear in space on the machine *Schonhage* [1]. Such series are used for the calculation of some mathematical constants of analysis and of the values of elementary functions at rational points.

$Sch(FQLIN - TIME // LIN - SPACE)$ will be used to denote the class of algorithms which are computable on Schonhage and are quasi-linear in time and linear in space. The main feature of Schonhage is its ability to execute recursive calls of procedures. Quasi-linear means that the complexity function is bounded by $O(n\log(n)^k)$ for some k .

The main advantage of algorithms based on series expansions is the relative simplicity of both the algorithms and the analysis of their computational complexity. Besides we can compute all the most commonly used constants of analysis using series expansions. For calculations with a small number of digits after the binary (or decimal) point, series are more efficient than other methods because of the small constants in estimations of their computational complexity. Therefore such algorithms are important in computer science for practical applications.

It is known [2] that linearly convergent hypergeometric series with rational coefficients can be calculated using the binary splitting method with time complexity $O(M(n)(\log(n))^2)$ and space complexity $O(n\log(n))$ (where $M(n)$ denotes the complexity of multiplication of n -bit integers). In recent publications, for example [3], algorithms based on a modified binary splitting method for the evaluation of linearly convergent hypergeometric series with time complexity $O(M(n)(\log(n))^2)$ and space complexity $O(n)$ are described.

In this paper we propose an algorithm for the evaluation of the hypergeometric series which is simpler in its practical implementation than the algorithm from [3]; the proposed algorithm is also quasi-linear in time and linear in space. The idea of working with a given accuracy to calculate the value of the hypergeometric series with a linear space complexity can be found at <http://numbers.computation.free.fr/Constants/Algorithms/splitting.html>.

The computational complexity of the constructive real numbers and functions is discussed in detail in [4]. The set of constructive real numbers with quasi-linear time and linear space complexity of calculating their dyadic approximations will be denoted by $Sch(FQLIN - TIME // LIN - SPACE)_{CF}$ (CF is the Cauchy function).

From now on, n will denote the length of the record of accuracy 2^{-n} of dyadic rational approximations. We will use $\log(k)$ for logarithms base 2.

1. Binary splitting method. This method is used to calculate the values of linearly convergent series with rational coefficients, in particular to calculate the hypergeometric series of the form

$$S = \sum_{i=0}^{\infty} \frac{a(i)}{b(i)} \prod_{j=0}^i \frac{p(j)}{q(j)}, \quad (1)$$

where a , b , p , and q are polynomials with integer coefficients; this series is linearly convergent if its partial sum

$$S(\mu(k)) = \sum_{i=0}^{\mu(k)} \frac{a(i)}{b(i)} \prod_{j=0}^i \frac{p(j)}{q(j)}, \quad (2)$$

where $\mu(k)$ is a linear function of k , differs from the exact value by not more than 2^{-k} : $|S - S(\mu(k))| \leq 2^{-k}$.

In its classical variant the binary splitting method works as follows. Put $k_1 = \mu(k)$. We consider the partial sum (2) for some integers i_1 and i_2 , $0 \leq i_1 \leq k_1$, $0 \leq i_2 \leq k_1$, $i_1 \leq i_2$:

$$S(i_1, i_2) = \sum_{i=i_1}^{i_2} \frac{a(i)p(i_1) \dots p(i)}{b(i)q(i_1) \dots q(i)}. \quad (3)$$

We calculate

$$\begin{aligned} P(i_1, i_2) &= p(i_1) \dots p(i_2), & Q(i_1, i_2) &= q(i_1) \dots q(i_2), \\ B(i_1, i_2) &= b(i_1) \dots b(i_2), & \text{and } T(i_1, i_2) &= B(i_1, i_2)Q(i_1, i_2)S(i_1, i_2). \end{aligned}$$

If $i_1 = i_2$, then these values are calculated directly. Otherwise, the series is divided into two parts, left and right, and $P(i_1, i_2)$, $Q(i_1, i_2)$, and $B(i_1, i_2)$ are calculated for each part recursively. Then the values obtained are combined:

$$\begin{aligned} P(i_1, i_2) &= P_l P_r, & Q(i_1, i_2) &= Q_l Q_r, & B(i_1, i_2) &= B_l B_r, \\ T(i_1, i_2) &= B_r Q_r T_l + B_l P_l T_r. \end{aligned} \quad (4)$$

The algorithm starts with $i_1 = 0$, $i_2 = k_1$. After calculating $T(0, k_1)$, $B(0, k_1)$, and $Q(0, k_1)$, we divide $T(0, k_1)$ by $B(0, k_1)Q(0, k_1)$ to get the result with the given accuracy. The lengths of $T(0, k_1)$ and $B(0, k_1)Q(0, k_1)$ are proportional to $k \log(k)$; therefore the binary splitting algorithm is quasi-linear in space; the time complexity of this algorithm is $O(M(k) \log(k)^2)$ [2].

2. The basic algorithm of class $Sch(FQLIN - TIME // LIN - SPACE)$.

We will modify the binary splitting method for evaluation of the hypergeometric series so that the algorithm is simple and is in class $Sch(FQLIN - TIME // LIN - SPACE)$.

Let's suppose that we want to calculate the values of the hypergeometric series (1) with an accuracy of 2^{-n} . It's enough to calculate the partial sum (2) with accuracy $2^{-(n+1)}$ because

$$\begin{aligned} |S - S(\mu(n+1))^*| &\leq |S - S(\mu(n+1))| + |S(\mu(n+1)) - S(\mu(n+1))^*| \\ &\leq 2^{-(n+1)} + 2^{-(n+1)} = 2^{-n}; \end{aligned}$$

here $S(\mu(n+1))^*$ is an approximate value of $S(\mu(n+1))$. Put $r = \mu(n+1)$. Take the minimum value k_1 such that $2^{k_1} \geq r$; let $r_1 = \lceil \frac{r}{k_1} \rceil$. We write the partial sum (2) as follows:

$$P(r) = \sigma_1 + \tau_2[\sigma_2 + \tau_3[\sigma_3 + \dots + \tau_{k_1-1}[\sigma_{k_1-1} + \tau_{k_1}\sigma_{k_1}]]], \quad (5)$$

where

$$\begin{aligned} \sigma_1 &= \sum_{i=0}^{r_1-1} \frac{a(i)}{b(i)} \prod_{j=0}^i \frac{p(j)}{q(j)} = \sigma'_1, \\ \tau_2 &= \prod_{j=0}^{r_1} \frac{p(j)}{q(j)}, \quad \sigma_2 = \frac{a(r_1)}{b(r_1)} + \sum_{i=r_1+1}^{2r_1-1} \frac{a(i)}{b(i)} \prod_{j=r_1+1}^i \frac{p(j)}{q(j)} = \xi_2 + \sigma'_2, \\ \tau_3 &= \prod_{j=r_1+1}^{2r_1} \frac{p(j)}{q(j)}, \quad \sigma_3 = \frac{a(2r_1)}{b(2r_1)} + \sum_{i=2r_1+1}^{3r_1-1} \frac{a(i)}{b(i)} \prod_{j=2r_1+1}^i \frac{p(j)}{q(j)} = \xi_3 + \sigma'_3, \\ &\dots \end{aligned}$$

The σ'_t are calculated by the usual binary splitting method for the sum (3), where $i_1 = (t-1)r_1 + 1$, $i_2 = t \cdot r_1 - 1$; for σ'_1 we take $i_1 = 0$, $i_2 = r_1 - 1$.

We introduce the notation

$$\begin{aligned} \omega &= l(W) + 1, \quad W = \max(A, B), \\ \text{where } A &= \max_{i=0..r} (|a(i)|, |b(i)|), \quad B = \max_{j=0..r} (|p(j)|, |q(j)|) \end{aligned}$$

(here, $l(u)$ is the length of the bit representation of u). Let's obtain an estimate of the computational complexity of the calculation of σ_t and τ_t .

Lemma (1). *The time complexity of the binary splitting method for calculation of σ_t is bounded by $O(M(r) \log(r))$; the space complexity is bounded by $O(r)$.*

Proof. Consider an arbitrary maximal chain of recursive calls derived from calculations by Formulas (4). Suppose that the numbering in the chain begins with its deepest element: $i = 1, \dots, \varsigma$, where ς is the length of the chain, $\varsigma \leq \lceil \log(2r_1) \rceil$.

Using mathematical induction on i , we show that the length of the representation of T at level i satisfies $l(T_i) < 2^{i+1}\omega + 2^i$. Note that $l(P_i) \leq 2^i\omega$, $l(Q_i) \leq 2^i\omega$, and $l(B_i) \leq 2^i\omega$ because there is a doubling of the length of the number representation when i increases. The induction begins with $i = 1$: $l(T_1) \leq 2\omega < 2^2\omega + 2^1$; next, the induction step is

$$l(T_{i+1}) < 2^i\omega + 2^i\omega + 2^{i+1}\omega + 2^i + 1 < 2^{(i+1)+1}\omega + 2^{i+1}.$$

Note that, based on this inequality, T_ς has length $O(r)$ because

$$l(T_\varsigma) < C_1 2^\varsigma \omega \leq C_2 \frac{r}{\log(r)} \log(r) = C_2 r;$$

here we take into account the fact that all coefficients $a(i)$, $b(i)$, $p(j)$, $q(j)$ are polynomials.

Let's estimate the time complexity of the calculation of σ'_t . We must take into account the property of the complexity of integer multiplication: $2M(2^{-1}m) \leq M(m)$ (quasi-linear and polynomial functions satisfy this property). Because at a tree node of recursive calls at level i there are C_3 multiplications of $2^{\varsigma-i+1}$ numbers of length at most $2^{i+1}\omega + 2^i$, the following estimate of the number of operations required to calculate σ'_t holds:

$$\begin{aligned} \text{Time}(\sigma'_t) &\leq C_3 \sum_{i=1}^{\varsigma} 2^{\varsigma-i} M(2^{i+1}\omega + 2^i) < C_4 \sum_{i=1}^{\varsigma} 2^{\varsigma-i} M(2^{i+2}\omega) \\ &= C_4 \sum_{i=1}^{\varsigma} 2^{\varsigma-i} M(2^{\varsigma-(\varsigma-(i+2))}\omega) \leq C_5 \sum_{i=1}^{\varsigma} 2^{\varsigma-i} 2^{-\varsigma+(i+2)} M(2^\varsigma \omega) \\ &\leq C_6 \varsigma M(r) \leq C_7 \log(r) M(r) \end{aligned}$$

(here we use the inequality for $2^\varsigma \omega$ from the estimate of $l(T_\varsigma)$). Final division gives $O(M(r))$ operations.

Now we estimate the space complexity of the computation of σ'_t . Because at a chain element at level i the amount of memory used for temporary variables is $C_8(2^{i+1}\omega + 2^i)$, the amount of memory in all simultaneously existing recursive calls is estimated as follows:

$$\text{Space}(\sigma'_t) \leq \sum_{i=1}^{\varsigma} C_8(2^{i+1}\omega + 2^i) \leq C_9(2^\varsigma \omega + 2^\varsigma) \leq C_{10}r = O(r).$$

□

Lemma (2). *The time complexity of the binary splitting method for the calculation of τ_t is bounded by $O(M(r) \log(r))$; its space complexity is bounded by $O(r)$.*

Proof. The estimation of the computational complexity of τ_t is the same as the estimation for σ_t due to the fact that the inequalities in the proof of lemma (1) are also suitable for τ_t (we can calculate the numerator and denominator of σ_t using the binary splitting method for products). □

We will calculate approximate values $P(r)^*$ with accuracy $2^{-(n+1)}$ in accordance with Formula (5) using the following iterative process:

$$\begin{aligned} h_1(m) &= \sigma_{k_1}^*, \\ \widehat{h}_i(m) &= \sigma_{k_1-i+1}^* + \tau_{k_1-i+2}^* h_{i-1}, \quad i = 1, \dots, k_1, \\ h_i(m) &= \widehat{h}_i(m) + \varepsilon_i; \end{aligned} \tag{6}$$

for $i = k_1$ we suppose $P(r)^* = h_{k_1}(m)$. Here $m \geq r$ (m will be chosen later), σ_i^* and τ_i^* are approximations of σ_i and τ_i with accuracy 2^{-m} . The values $h_i(m)$ are obtained by discarding bits $q_{m+1}q_{m+2} \dots q_{m+j}$ of numbers $\widehat{h}_i(m)$ after the binary point starting with the $(m+1)$ bit:

$$|\varepsilon_i| = |h_i(m) - \widehat{h}_i(m)| = 0.0 \dots 0 q_{m+1} q_{m+2} \dots q_{m+j}, \tag{7}$$

and the sign of ϵ_i is the same as the sign of $\widehat{h}_i(m)$ (it is clear that $|\epsilon_i| < 2^{-m}$).

Let's assume that the following conditions hold:

$$b(i) \geq 2 \text{ for all } i, \quad \frac{p(j)}{q(j)} \leq 1 \text{ for all } j. \quad (8)$$

We prove the following two lemmas.

Lemma (3). *For every $i \in 1..k_1$*

$$|h_i(m)| < (i+1)r_1W. \quad (9)$$

Proof. We apply induction on j to $h_j(m)$. The induction base is $j = 1$: $|h_1(m)| \leq \frac{1}{2}r_1W + 2^{-m} < 2r_1W$. The induction step is $(j+1) \geq 2$:

$$\begin{aligned} |h_{j+1}(m)| &= |\sigma_{k_1-(j+1)+1}^* + \tau_{k_1-(j+1)+2}^* h_j + \epsilon_{j+1}| \\ &\leq \frac{1}{2}r_1W + (j+1)r_1W + 2^{-m} < ((j+1)+1)r_1W. \end{aligned}$$

□

Lemma (4). *The error of calculation of $h_{k_1}(m)$ according to scheme (6) is estimated as*

$$\Delta(k_1, m) < 2^{-m}mk_1^2W.$$

Proof. Let's denote

$$H_1 = \sigma_{k_1}, \quad H_i = \sigma_{k_1-i+1} + \tau_{k_1-i+2}H_{i-1}, \quad \eta(i, m) = |h_i(m) - H_i|.$$

We use the method of mathematical induction for $\eta(j, m)$ for j . The induction base is $j = 1$:

$$\eta(1, m) = |h_1(m) - H_1| = |\sigma_{k_1}^* - \sigma_{k_1}| < 2^{-m}.$$

The induction step is $(j+1) \geq 2$:

$$\begin{aligned} \eta(j+1, m) &= |\sigma_{k_1-(j+1)+1}^* + \tau_{k_1-(j+1)+2}^* h_j(m) + \epsilon_{j+1} - \\ &\quad \sigma_{k_1-(j+1)+1} - \tau_{k_1-(j+1)+2}H_j| \\ &< |\tau_v^* h_j(m) - \tau_v h_j(m) + \tau_v h_j(m) - \tau_v H_j| + 2 \cdot 2^{-m} \\ &\leq 2^{-m}h_j(m) + \tau_v \eta(j, m) + 2 \cdot 2^{-m}. \end{aligned}$$

Because of (9) $|h_j(m)| < (j+1)r_1W$ and, by the induction hypothesis, $\eta(j, m) < 2^{-m}mj^2W$, we get

$$\begin{aligned} \eta(j+1, m) &< 2^{-m}(j+1)r_1W + 2^{-m}mj^2W + 2 \cdot 2^{-m} \\ &< 2^{-m}(j+1)mW + 2^{-m}m(j+1)^2W = 2^{-m}m(j+2)^2W. \end{aligned}$$

From $\Delta(k_1, m) = \eta(k_1, m)$ we get the required inequality. □

Lemma 4 implies that to compute $P(r)^*$ with accuracy $2^{-(n+1)}$ it suffices to take m such that the following holds

$$m \geq (n+1) + \lceil \log(n+1) + 2\log(r) + \log(W) \rceil, \quad (10)$$

We denote the algorithm for the calculation of the hypergeometric series which uses scheme (6) as *LinSpaceBinSplit* (binary splitting method with linear space complexity).

Algorithm *LinSpaceBinSplit*. **The approximate value of the hypergeometric series.**

Input: Record of the accuracy 2^{-n} .

Output: The approximate value of (1) with accuracy 2^{-n} .

Description:

Table 1: Formulas for evaluation of constants

Constant	Formula	Series
e	$e = \exp(1)$	$\exp(1) = 2 \sum_{i=0}^{\infty} \frac{1}{2^i i!}$
π	$\pi = 16\alpha - 4\beta$, $\alpha = \operatorname{arctg} \frac{1}{5}$, $\beta = \operatorname{arctg} \frac{1}{239}$	$\operatorname{arctg} \left(\frac{1}{5}\right) = 2 \frac{1}{5} \sum_{i=0}^{\infty} (-1)^i \frac{1}{2(2i+1)5^{2i}}$, $\operatorname{arctg} \left(\frac{1}{239}\right) = 2 \frac{1}{239} \sum_{i=0}^{\infty} (-1)^i \frac{1}{2(2i+1)239^{2i}}$
$\zeta(3)$		$\sum_{i=0}^{\infty} \frac{(-1)^i (205i^2 + 250i + 77) ((i+1)!)^5 (i!)^5}{2((2i+2)!)^5}$

Table 2: Series for calculations of constants

Constant	a(i)	b(i)	p(j)	q(j)
e	1	2	1	j
π	1 1	$2(2i+1)$ $2(2i+1)$	-1 -1	5^2 239^2
$\zeta(3)$	$205i^2 + 250i + 77$	2	$p(0) = 1$, $p(j) = -j^5$	$32(j+1)^5$

- 1) compute $r := \mu(n+1)$; choose k_1 so that $2^{k_1} \geq r$; compute $r_1 := \lceil \frac{r}{k_1} \rceil$; compute W ;
- 2) find m using Formula (10);
- 3) $h := \sigma_{k_1}^*$ (using the usual binary splitting method with accuracy 2^{-m});
- 4) make loops through i from 2 to k_1 :
 - a) calculate $v_1 := \sigma_{k_1-i+1}^*$ with accuracy 2^{-m} using the usual binary splitting method and $v_2 := \tau_{k_1-i+2}^*$ with accuracy 2^{-m} ,
 - b) calculate the expression $\hat{h} := v_1 + v_2 h$,
 - c) assign value \hat{h} rounded in accordance with (7) to h ;
- 5) write h to output.

We estimate the time computational complexity of the algorithm taking into account that r and m are linearly dependent on n :

- $O(\log(n))$ computations of σ_t gives $O(M(n) \log(n)^2)$;
- $O(\log(n))$ computations of τ_t gives $O(M(n) \log(n)^2)$;
- $O(\log(n))$ multiplications of numbers of the length $O(n)$ gives $O(M(n) \log(n))$;

in total we obtain $O(M(n) \log(n)^2)$ bit operations. The space complexity of algorithm *LinSpaceBinSplit* is $O(n)$ because in all calculations in this algorithm numbers of length $O(n)$ are used.

Theorem. *The modified binary splitting algorithm for the calculation of the hypergeometric series, *LinSpaceBinSplit*, belongs to $Sch(FQLIN - TIME // LIN - SPACE)$.*

Conclusion. In Tables 1 and 2 there are formulas and series for the calculation of some frequently used constants of mathematical analysis. These series converge linearly (see [2, 3]) and they satisfy conditions (8); hence, to calculate them, we can use *LinSpaceBinSplit* and therefore these constants belong to the set of constructive real numbers $Sch(FQLIN - TIME // LIN - SPACE)_{CF}$. The algorithm *LinSpaceBinSplit* can also be used to calculate approximations to many other constants and to the values of elementary functions at rational points.

If we use the Schonhage–Strassen algorithm for integer multiplication with a time complexity of $O(n \log(n) \log \log(n))$, then the time complexity of *LinSpaceBinSplit* will be $O(n \log(n)^3 \log \log(n))$; when we use a simple recursive method for integer multiplication with a time complexity of $O(n^{\log(3)})$, the time complexity of *LinSpaceBinSplit* will be $O(n^{\log(3)} \log(n)^2)$.

Let's note that the series of the constant e converges with the rate $2^{-O(n \log(n))}$, so the time complexity of the calculation of e using *LinSpaceBinSplit* will be $O(M(n) \log(n))$ and its space complexity will be $O\left(\frac{n}{\log(n)}\right)$.

References

- [1] Schonhage A., Grotefeld A.F.W, Vetter E. *Fast Algorithms. A Multitape Turing Machine Implementation.* // Germany: Brockhaus, 1994.
- [2] Haible B., Papanikolaou T. Fast multiple-precision evaluation of series of rational numbers // *Proc. of the Third Intern. Symposium on Algorithmic Number Theory.* June 21–25, 1998. pp. 338–350.
- [3] Cheng H., Gergel B., Kim E., Zima E. Space-efficient evaluation of hypergeometric series // *ACM SIGSAM Bull. Communications in Computer Algebra.* 2005. Vol. 39, No. 2. pp. 41–52.
- [4] Ko K. *Complexity Theory of Real Functions.* // Boston: Birkhauser, 1991.